МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе №6**

**ИЗБЫТОЧНОЕ КОДИРОВАНИЕ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ. ЦИКЛИЧЕСКИЕ КОДЫ**

Выполнил:

Cтудент 3 курса 1 группы

Парибок И. А.

Вариант 5

Минск 2022

Цель: приобретение практических навыков кодирования/декодирования двоичных данных при использовании циклических кодов (ЦК).

**Практическое задание**  
 1. Задание выполняется по указанию преподавателя в соответствии с вариантом из табл. 6.2, из которого выбирается порождающий полином ЦК, а по значению соответствующего ему значения *r* – длина *k* информационного слова *X*k. Полагаем, что каждый полином соответствует коду, обнаруживающему и исправляющему одиночные ошибки в кодовых словах. Определить параметры (*n*, *k*)-кода для своего варианта. Основой задания является разработка приложения.  
 2. Составить порождающую матрицу (n, k)-кода в соответствии с формулой (6.7), трансформировать ее в каноническую форму и далее – в проверочную матрицу канонической формы.  
 3. Используя порождающую матрицу ЦК, вычислить избыточные символы (слово *X*r) кодового слова *X*n и сформировать это кодовое слово.

4. Закрепить теоретические знания по алгебраическому описанию и использованию ЦК для повышения надежности передачи и хранения в памяти компьютера двоичных данных, для контроля интегральности файлов информации.

5. Разработать приложение для кодирования/декодирования двоичной информации циклическим кодом.

6. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Выполнение работы:**



Рисунок 1 – ­Варианты заданий

Результаты выполнения задания представлены в рисунках 2-6. Листинг программы изложен в приложении А.

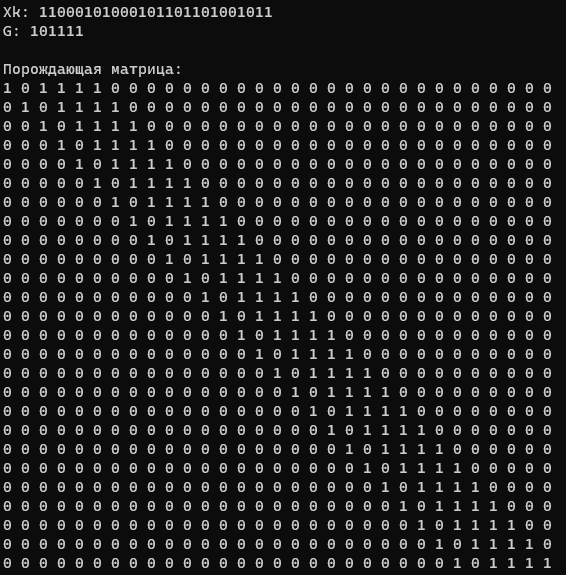


Рисунок 2 – Информационное слово, полином и порождающая матрица

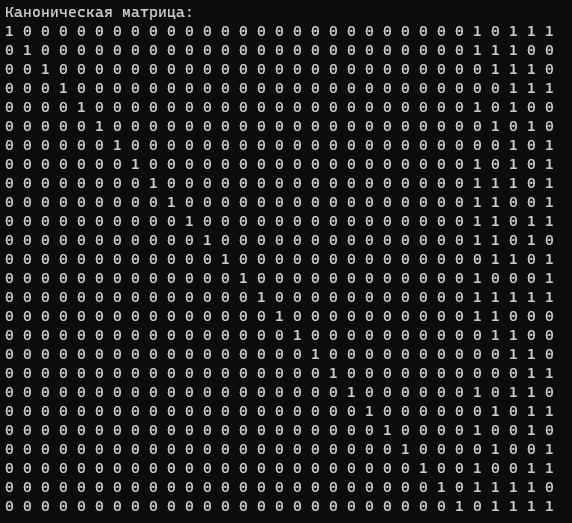


Рисунок 3 – Каноническая матрица

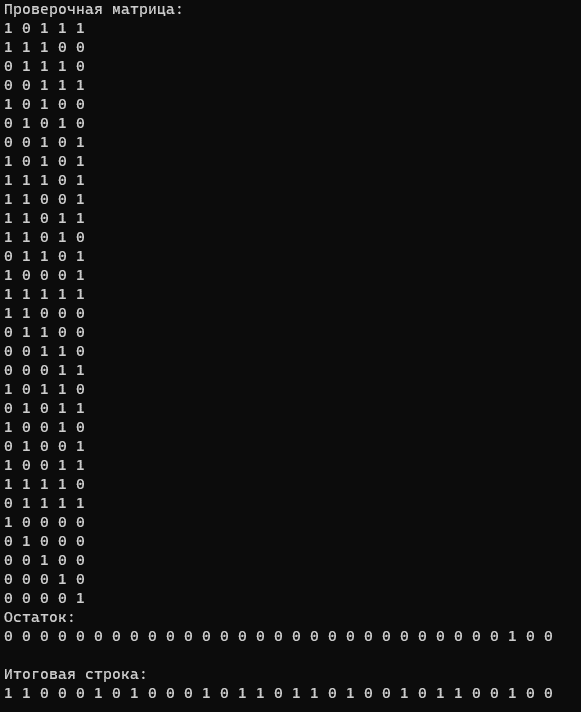


Рисунок 4 – Проверочная матрица, остаток от деления и итоговая строка

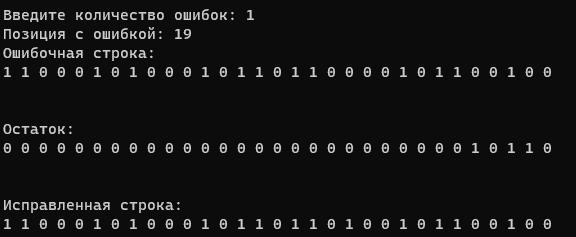


Рисунок 5 – Результат исправления одной ошибки

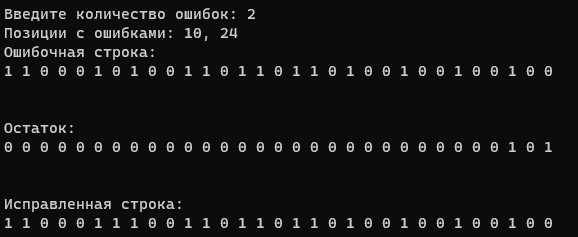


Рисунок 6 – Результат исправления двух ошибок

Вывод: в результате данной лабораторной работы было разработано приложение для кодирования/декодирования двоичной информации циклическим кодом.

**Приложение А**

using System;

namespace Lab6

{

class Program

{

static void Main(string[] args)

{

string Xk = "11000101000101101101001011";

string g = "101111";

Random rnd = new Random();

int pos, pos2;

int r = 5;

int k = Xk.Length;

int n = r + k;

int error;

int[] masXk = new int[k];

StrInMas(masXk, Xk);

int[] masXr = new int[g.Length];

StrInMas(masXr, g);

Console.WriteLine("Xk: " + Xk);

Console.WriteLine("G: " + g);

int[,] generationMatrix = new int[k, n];

CreateGenerationMatrix(generationMatrix, masXr, k, n);

Console.WriteLine("\nПорождающая матрица:");

OutMatrix(generationMatrix, k, n);

CreateCanonicalMatrix(generationMatrix, k, n);

Console.WriteLine("\n\nКаноническая матрица:");

OutMatrix(generationMatrix, k, n);

int[,] checkMatrix = new int[n, r];

CreateCheckMatrix(checkMatrix, generationMatrix, k, n);

Console.WriteLine("\nПроверочная матрица:");

OutMatrix(checkMatrix, n, r);

int[] masXn = new int[n];

Shift(masXn, masXk, r);

SearchResidue(masXn, masXr);

Console.WriteLine("Остаток:");

OutMass(masXn);

Console.WriteLine("Итоговая строка:");

Shift(masXn, masXk, r);

OutMass(masXn);

Console.Write("Введите количество ошибок: ");

int errCount = Convert.ToInt32(Console.ReadLine());

if (errCount == 1)

{

pos = rnd.Next(0, masXn.Length - 1);

if (masXn[pos] == 1) masXn[pos] = 0;

else masXn[pos] = 1;

Console.WriteLine($"Позиция с ошибкой: {pos}");

}

if (errCount == 2)

{

pos = rnd.Next(0, masXn.Length - 1);

if (masXn[pos] == 1) masXn[pos] = 0;

else masXn[pos] = 1;

pos2 = rnd.Next(0, masXn.Length - 1);

while (pos == pos2) pos2 = rnd.Next(0, masXn.Length);

if (masXn[pos2] == 1) masXn[pos2] = 0;

else masXn[pos2] = 1;

Console.WriteLine($"Позиции с ошибками: {pos}, {pos2}");

}

if (errCount > 2)

{

Console.WriteLine("Много ошибок");

return;

}

Console.WriteLine("Ошибочная строка:");

OutMass(masXn);

SearchError(masXn, masXr, checkMatrix, r);

Console.ReadKey();

}

#region Functions

public static int[] SearchError(int[] masXn, int[] masXr, int[,] checkMatrix, int r)

{

int n = masXn.Length;

int k = n - r;

int[] masXnSecond = new int[n];

for (int i = 0; i < n; i++)

{

masXnSecond[i] = masXn[i];

}

SearchResidue(masXnSecond, masXr);

Console.WriteLine("\nОстаток:");

OutMass(masXnSecond);

for (int i = 0; i < n; i++)

{

int coincidence = 0;

for (int j = 0; j < r; j++)

{

if (checkMatrix[i, j] == masXnSecond[k + j])

{

coincidence++;

}

}

if (coincidence == r)

{

masXn[i] = (masXn[i] + 1) % 2;

break;

}

}

Console.WriteLine("\nИсправленная строка:");

OutMass(masXn);

return masXn;

}

public static int[] SearchResidue(int[] masXn, int[] masXr)

{

int end = masXn.Length - masXr.Length + 1;

for (int i = 0; i < end; i++)

{

if (masXn[i] == 1)

{

AddingMasMod2(masXn, masXr, i);

//OutMass(masXn);

}

}

return masXn;

}

//Сложение массивов по модулю 2 с опр. позиции

public static int[] AddingMasMod2(int[] mas1, int[] mas2, int pos)

{

int end = pos + mas2.Length;

for (int i = pos; i < end; i++)

{

mas1[i] = (mas1[i] + mas2[i - pos]) % 2;

}

return mas1;

}

//Смещение на массива r

public static int[] Shift(int[] shiftMas, int[] mas, int r)

{

for (int i = 0; i < mas.Length; i++)

{

shiftMas[i] = mas[i];

}

return shiftMas;

}

//Преобразование сторки в массив

public static int[] StrInMas(int[] mas, string str)

{

for (int i = 0; i < str.Length; i++)

{

if (str[i] == 49)

mas[i] = 1;

else mas[i] = 0;

}

return mas;

}

//Создание Порождающей матрицы

static int[,] CreateGenerationMatrix(int[,] generationMatrix, int[] mas, int k, int n)

{

//Заполняем первую строку в проверочной матрице

for (int i = 0; i < n; i++)

{

if (i < mas.Length)

{

generationMatrix[0, i] = mas[i];

}

else

{

generationMatrix[0, i] = 0;

}

}

//Сдвигаем каждую строки вправо от предыдущей

for (int i = 1; i < k; i++)

{

for (int j = 0; j < n - 1; j++)

{

generationMatrix[i, j + 1] = generationMatrix[i - 1, j];

}

generationMatrix[i, 0] = generationMatrix[i - 1, n - 1];

}

return generationMatrix;

}

//Приведение порождающей матрицы к каноническому виду

static int[,] CreateCanonicalMatrix(int[,] generationMatrix, int k, int n)

{

//Перебираем строки для преведению к каноническому виду

for (int i = 0; i < k; i++)

{

int i2 = i + 1;

//Перебираем элементы строки, но только до k-элемента

for (int j = i + 1; j < k; j++)

{

//если мы нашли единицу в строке, то...

if (generationMatrix[i, j] == 1)

{

//перебираем этот столбец, пока не найдем единицу

for (; i2 < k; i2++)

{

bool repeat = false;

//Если нашли, то складываем обе строки

if (generationMatrix[i2, j] == 1)

{

for (int j2 = j - 1; j2 > 0; j2--)

{

//Проверяем, есть ли до этой 1 еще 1, если есть то эту строку пропускаем

if (generationMatrix[i2, j2] == 1)

{

repeat = true;

}

}

if (repeat)

continue;

AddingLinesMatrixMod2(generationMatrix, i, i2, n);

i2++;

break;

}

}

}

}

}

return generationMatrix;

}

//Преобразование канонической матрицы в проверочную

static int[,] CreateCheckMatrix(int[,] checkMatrix, int[,] generationMatrix, int k, int n)

{

int r = n - k;

for (int i = 0; i < k; i++)

{

for (int j = 0; j < r; j++)

{

checkMatrix[i, j] = generationMatrix[i, k + j];

}

}

for (int i = k; i < n; i++)

{

for (int j = 0; j < r; j++)

{

if (j == i - k)

{

checkMatrix[i, j] = 1;

}

else

{

checkMatrix[i, j] = 0;

}

}

}

return checkMatrix;

}

//Сложение строк матрицы

public static int[,] AddingLinesMatrixMod2(int[,] matrix, int str1, int str2, int lengthString)

{

//Console.WriteLine(str1 + " и " + str2);

for (int i = 0; i < lengthString; i++)

{

matrix[str1, i] = (matrix[str1, i] + matrix[str2, i]) % 2;

}

return matrix;

}

//вывод матрицы

public static void OutMatrix(int[,] matrix, int k, int n)

{

for (int i = 0; i < k; i++)

{

for (int j = 0; j < n; j++)

{

Console.Write(matrix[i, j] + " ");

//if (j + 1 == k) Console.Write("|");

}

Console.WriteLine();

}

}

//вывод одномерного массива

public static void OutMass(int[] mas)

{

for (int i = 0; i < mas.Length; i++)

{

//if (i == k) Console.Write("|");

Console.Write(mas[i] + " ");

}

Console.WriteLine("\n");

}

#endregion

}

}